Shibboleth Relying Party Configuration GUI

Background

One if the traditional impediments to smaller school adoption of the Shibboleth IdP has been its XML-based configuration management. Recent work by the Shibboleth team for TIER has expanded the ability to control the IdP configuration via entity attributes ("tags") associated with the SAML metadata of a relying party. These enhancements will become part of the Shibboleth distribution with the next release. The goal for the work has been to enable metadata-based control of 80% or more of the routine day-to-day configuration changes made by administrators.

The Shibboleth enhancements were designed to facilitate the implementation of a GUI to manage configuration while, to the greatest extent possible, minimize the interactions between the GUI and Shibboleth itself. The GUI's only required role is to manage metadata filters embedded within the metadata-providers.xml file to apply designated entity attributes at runtime. An optional set of features would encompass more complete management of metadata sources as well as the associated filters.

The Shibboleth documentation on this feature in the form in which it will be made available in V3.4 is here.

Application Requirements

The GUI is expected to be a simple web application that can be trivially deployed into many environments. One required deployment scenario is the TIER Shibboleth VM appliance itself. Schools may choose other local or cloud deployments. Basic application functionality includes:

1. Provide a mechanism to create new and edit existing entity attributes:
    a. Display a screen containing all supported attributes.
    b. In the case where an attribute is already defined, maintain the possible values.
    c. Provide an organized display enabling easy selection/deselection of features/attributes by name. Provide a simple description of functionality for each item listed.
2. Provide mechanisms to create, view, edit, delete, and duplicate associations between supported entity attributes and the entities to which they should be associated. Associations should be possible on the basis of at least:
    a. Specific relying parties.
    b. Ad-hoc collections of specific relying parties.
    c. Relying parties that possess existing entity attribute name/value pairs (e.g. entity categories such as the Research and Scholarship designation).
    d. Optionally, relying parties for which a Java scriptlet evaluates successfully.
3. Provide mechanisms to import and export associations.
4. Provide a mechanism to generate the appropriate filters and update the metadata-providers.xml file:
    a. Sanity check the contents as part of the generation process.
    b. Archive the previous version of the file.
    c. Enable review before publishing.
5. Maintain an optional comments section for each set of associations.
6. Maintain a log all transactions.
7. Retain some small number of previous versions of the configuration for each association.
8. Support common templates of collections of entity attribute properties to apply, and the ability to attach such templates by reference or value (the former maintaining the link such that changes to the template automatically apply to each association of the template).
9. Simple user authorization support is required. Support for REMOTE_USER matched against a deployer-maintained file or table of authorized users is sufficient.
10. Software updates must be trivially deployable while retaining existing configurations.
11. Internationalization of the UI must be possible, but actual translations are not required.
12. Data integrity in the face of multiple users interacting with the application is required.
13. Ability to coexist with other school-developed filters; the tool must insert its configuration into specific placeholder(s) in the metadata-providers.xml file.

Technical Infrastructure Preferences

The technical infrastructure should be designed to minimize deployment and operational requirements. The application will be used occasionally as opposed to continually, so trade-offs to optimize ease of deployment and operation over anything related to performance are appropriate.

1. Ease of Configuration
    a. The addition of new entity attribute properties should not require code changes
    b. Changes to UI language should not require code changes, and ideally should be based on standard HTTP language negotiation.
    c. Simple platform requirements and tooling is desired, specifics subject to approval by TIER. Components beyond those already required by the Shibboleth IdP should be minimized.
2. Ease of Operation
    a. The decision to use a database or a simple filesystem scheme should be part of the design document. The goal is to minimize, within reason, operational needs for schools running the application.
    b. Basic installation and upgrade support and automation are required.
    c. Deployment and, if needed, usage documentation are required.
    d. Security
        i. Even though deployments are expected to be private and not public facing, programming to protect against standard web application security risks is required.

Development Support

1. TIER Shibboleth VM distribution for testing
    a. A standard TIER appliance running the updated Shibboleth code
2. An Initial entity attribute set based on the convention and supported properties outlined in the documentation.

3. Handcrafted example metadata filters demonstrating how to express the required associations.
4. Assistance, as needed, with any changes required by the incorporation of the TIER modifications into the Shibboleth distribution.
5. Assistance with testing.